#19
B. Board
6/12/01

Docket No.: SJO-96-505 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Mark E. Davis, et al.

Serial No. 08/799,073

Filed: February 11, 1997

For: STREAMING COMPUTER SYSTEM AND METHOD WITH MULTI-VERSION PROTOCOL COMPATIBILITY

Examiner: T. Vu

Group Art Unit: 2152

Appeal No.: _____

BRIEF OF APPELLANTS

Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

In accordance with 37 C.F.R. §1.192, Appellants hereby submit the Appellants' Brief on Appeal (in triplicate) from the final rejection of claims 1-8, 10-21, and 23-32 of the above-identified application, as set forth in the Office Action mailed December 21, 2000.

Please charge the amount of $310 to cover the required fee for filing this Appeal Brief as set forth under 37 C.F.R. §1.17(f) to Deposit Account No. 09-0460 of IBM Corporation, the assignee of the present application. Also, please charge any additional fees or credit any overpayments to Deposit Account No. 09-0460.

I. REAL PARTY IN INTEREST

The real party in interest is the IBM Corporation, the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences for the above-reference patent application.

III. STATUS OF CLAIMS

Claims 1-8, 10-21, and 23-32 are pending in the application. These claims are shown in the Appendix attached to this Appeal Brief.

On October 15, 1998, a first Office Action was mailed. The first Office Action rejected claims 1-6, 11, 12, and 13-30 under 35 U.S.C. § 102 as being anticipated by United States Patent No. 5,598,276, issued to Cookson et al. Claims 7-8 were rejected under 35 U.S.C. § 103 as being unpatentable over Cookson and further in view of United States Patent No. 5,767,894, issued to Fuller et al. Claims 9-10 were rejected under 35 U.S.C. § 103 as being unpatentable over Cookson in view of Fuller et al. and further in view of Fielding et al.

On January 15, 1999, the Applicants filed Remarks in response to these rejections, leaving the claims unamended.

On March 29, 1999, a Final Office Action was mailed, maintaining the rejections of the first Office Action.

On May 28, 1999, the Applicants filed an Amendment under 37 C.F.R. § 1.116, amending claims 1, 13, 19, 24, 28, 29, and 30.

On June 22, 1999, an Advisory Action was mailed, refusing to enter the amendments because they would require an extended search.

On June 28, 1999, the Applicants filed a Continued Prosecution Application requesting that the unentered amendments be entered.

On August 30, 1999, a first Office Action was mailed. The first Office Action rejected claims 1-7, and 12 under 35 U.S.C. § 102(a)(e) as unpatentable over U.S. Patent 5,790,802, issued to Van Loon et al. Claims 8-11 were rejected under 35 U.S.C. § 103 as unpatentable over Van Loon in view of U.S. Patent No. 5,893,908, issued to Cullen et al. Claims 13-30 were rejected under analogous rationale to the rejections of claims 1-12.

On November 24, 1999, the Applicants filed an Amendment canceling claims 9 and 22, amending claims 1, 10, 19, and 29, and adding claims 31 and 32.

On February 18, 2000, a second Office Action was issued. The second Office Action rejected claims 1-8, 10-21, and 23-30 under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,790,802, issued to Heath et al. in view of U.S. Patent No. 5,959,543, issued to LaPorta et al. Claims 31 and 32 were indicated as allowable.

On May 17, 2000, the Applicants filed a communication leaving the claims unamended and arguing that the claims are patentable over the references cited in the second Office Action mailed February 18, 2000.

On August 15, 2000, a third Office Action was mailed. The third Office Action rejected claims 1-8, 10-21, and 23-32 as unpatentable over the earlier cited Van Loon reference in view of U.S. Patent No. 4,912,637, issued to Sheedy et al.

On November 15, 2000, the Applicants filed an Amendment amending claims 1 and 13 and arguing the claims are allowable over the cited references.

On December 21, 2000, a Final Office Action was mailed, maintaining the rejection of the third Office Action mailed August 15, 2000.

On March 21, 2001, the Applicants filed a Notice of Appeal.

IV. STATUS OF AMENDMENTS

A Notice of Appeal was filed in response to the final Office Action. No amendments have been made subsequent to the final Office Action.

V. SUMMARY OF THE INVENTION

Briefly, the Applicants' invention as described in independent claims 1, 13, 19, 24, 28, and 29 is generally directed to a method, apparatus and article of manufacture for transmitting and/or receiving a data segment in a data stream according to a selected one of a plurality of additive streaming protocols. A first stream of data according to a first version of the streaming protocol is outputted, and additional streams of data are sequentially appended to the first stream of data according to each subsequent version of the streaming protocol.

VI. ISSUES PRESENTED FOR REVIEW

Whether claims 1-8, 10-21, and 23-32 are unpatentable over the Van Loon reference in view of the Sheedy reference.

VII. GROUPING OF CLAIMS

The rejected claims do not stand or fall together. Each claim is independently patentable. Separate arguments for the patentability of each claim group are presented below.

VIII. ARGUMENTS

A. Art-Related Rejections

1. Rejections Under 35 U.S.C. § 103

In item (4) of the Final Office Action claims 1-8, 10-21, and 23-33 were rejected as being unpatentable over the Van Loon reference in view of the Sheedy reference.

2. The Van Loon Reference

U.S. Patent No. 5,790,802, issued to Van Loon et al on August 4, 1998 discloses a method for exchanging a message between systems in which the message comprises an information element of a first type and information element of a second type.

The Van Loon reference teaches that additive streaming protocols are known in the art. (See col. 1, lines 12-27). The Van Loon reference then teaches that such protocols are problematic, because they allow freedom in developing subsequent protocol versions (presumably, because they must be additive), and because messages based on higher protocol versions are essentially wasted information (e.g. "have no significance whatsoever") for systems operating with lower protocol versions. (See col. 1, lines 29-34).

The Van Loon reference therefore proceeds to disclose a system which is not limited to additive streaming protocols. (See col. 5, line 43, through col. 7, line 6). As the Applicants understand (perhaps because of translation problems, the Van Loon patent is by no means a model of clarity), the Van Loon reference discloses the following.

A message is transmitted having a header (formed, for example, by bytes A and B), an information element of a first type (formed, for example, by bytes C and D), an information element of the second type (formed, for example, by bytes E and F), and a subsequent information element of the second type (formed, for example, by bytes G and H). See, e.g. col. 5, lines 55-61.

When the message is received, a processor 10 analyzes either the header of the message (bytes A and B) or the remaining portion of the message (bytes C, D, E, and F), to generate a control signal. A memory means 13 stores a table having two columns; the first with information elements of the first type (corresponding to bytes A and B) and the second with information elements of the second type (corresponding to bytes C and D). The processor 10 and the memory 13 interact via the control link 18 and a comparison between data stored in the table and the incoming data is performed to determine the protocol. Presumably, because the new protocols can be described in the table, greater flexibility in designing new protocols is provided.

One of the confusing aspects of the Van Loon reference is that it does not disclose how the data from the different protocols are combined, or how they are handled after receipt to use the data with the required protocol. At best, all Van Loon discloses is the combination of data from one protocol with another. Much of what the Examiner suggests with response to how the Van Loon reference works is hindsight reconstruction based on conjecture. For example, the Office Action suggests that the step of:

> *sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol*

is taught at in the following portion of the Van Loon reference.

> The unit depicted in FIG. 3 consists of a system 35, of which a protocol-dependent section 36, based on one protocol version, is linked via a bus 37 to second device 38. Device 38 is further linked, via a bus 32, to a protocol-dependent subsystem 31, based on a first protocol version, of a system 30 and is further linked, via a bus 34, to a protocol-dependent subsystem 33, based on a second protocol version, of said system 30. If section (or subsystem) of a system is based on a first (lower) protocol version, this will result, for example, in said (sub)system despatching a message which comprises an information element of a first type, and if a section (or subsystem) of a system is based on a second (higher) protocol version, this will result, for example, in said (sub)system despatching a message which comprises an information element of a second type. In this situation, a message consists, for example, of eight bytes A, B, C, D, E, F, G and H, a header being formed by bytes A and B, an information element of a first type being formed by bytes C and D, an information element of a second type being formed by bytes E and F, and a subsequent information element of a second type being formed by bytes G and H.

The Examiner seems to be inferring that the input to the Van Loon translator includes only data according to a first protocol and the output includes data from a second protocol appended to data according to the first protocol (e.g. the device accepts "ABCD" and produces "ABCDEF"). However, the paragraphs that follow describe that the device (35) which does the translation from one protocol to another, accepts an input data stream including all characters A,B,C, D, E, F, G, and H."

> The mode of operation of device 38 in one direction is as follows. A message which comes from subsystem 31 and comprises an information element of a first type (bytes C and D) is fed, via bus 32, to buffer memory 41 of device 38. By means of a control signal via control link 48, processor 40 is informed of the arrival of said message, whereupon the message, via bus 46, is analysed by processor 40 (either by analysis of the header, bytes A and B, or by analysis of the content of the remaining portion of the message, bytes C, D, E, F, G and H).

Van Loon appears to substitute new bytes E and F it has looked up via a table for the original bytes E and F that were originally sent to it:

> Then the information element of a first type (bytes C and D) of the message is fed, via bus 46, to selection means 44 which, for example, store said information element (bytes C and D). In response to a control signal coming from processor 40 via control link 50. Thereafter, processor 40 is informed of the arrival of said information element (bytes C and D) in response to a control signal coming from processor 40 via control link 51, memory means 43 successively generate various information elements, stored in memory means 43, of a first type, which are supplied to selection means 44 via bus 47. To this end, memory means 43 are equipped, for example, with a table having two columns, the first column comprising information elements of a first type and the second column comprising information elements of a second type, where information the same situated on the same row, of a first and second type, respectively, have at least in part substantially the same information content. Selection means 44 compare each information element of a first type which has arrived via bus 46 with the stored information element of a first type (bytes C and D) and in case of identity inform processor 40 by means of a control signal via control link 50. In response thereto, processor 40 generates a subsequent control signal via control link 51 to memory means 43 which, in response thereto, generate that information element of a second type which is situated on the same row as the selected information element of a first type. Said selected information element of a second type is fed, via bus 46, to buffer memory 45 which stores said selected information element of a second type in response to a control signal coming from processor 40 and supplied via control link 52 (bytes E and F).

The format of the output of the device (35) is left unspecified, except to say that it includes all of the elements A, B, C, D, E, F, G, and H.

> Then the remaining portion of the message (bytes A, B, C, D, G and H) stored in buffer memory 41 is fed, via bus 46, to buffer memory 45 which stores said remaining portion of the message, whereupon the message which has thus been stored in buffer memory 45 and which now comprises both an information element of a first type (bytes C and D) and an information element of a second type (bytes E and F) can be despatched to system 35 via bus 37.

The Van Loon reference also does not disclose how the combined message, whether it is in the form ABCDEFGH, ABEFCDGH or otherwise, is interpreted by the device which is receiving the message (in this case, system 35), and therefore sheds no light on how the message is combined (if the buffer memory 41 were a FIFO (first in-first out) buffer, the order would seem to be

"EFABCDGH"). Significantly, Van Loon does not disclose where the information required to pull out the relevant portions of the message might be found or how this would be accomplished.

> Independently of the protocol version on which section 36 of system 35 is based, system 35 is now able to process this message without it having been necessary when the second protocol version was designed to take the first protocol version very much into account.

B. The Sheedy Reference

U.S. Patent No. 4,912,637, issued September 28, 1999 to Sheedy et al. discloses a system for preserving, generating, and merging different versions of a common module that utilizes a line file storing the text of every line in a version and addressing each line with a unique line identifier. Any desired version may be generated directly without creating intermediate versions. The unique line identifiers facilitate a merge operation that does not duplicate lines.

The teachings of Sheedy et al. particularly address a problem which occurs in software development where alterations may be separately made to the same module (or document) thereby creating two or more divergent version paths. The system of Sheedy et al. uses two basic files, a line file and a variant history file to facilitate version management. Within these files, change tags are used to track the various differences (embodied in different lines) between different versions.

1. Independent Claims 1, 19, and 29 are Patentable Over the Cited References

Even when combined, the Van Loon and Sheedy references do not disclose the following limitations of claim 1:

sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

delimiting the data segment in the data stream using begin and end tags.

As described above in Section VIII.A.2, the Van Loon reference does not disclose appending additional streams of data to the first stream of data if the selected version of the

streaming protocol is not the first version of the streaming protocol. The data translation devices 7 and 38 of the Van Loon reference accept bytes ABCDEFGH, where bytes C and D are according to a first protocol, and substitute bytes E and F (derived from bytes C and D and a predetermined table) according to a second protocol for the E and F bytes in the originally received message. Essentially, the Van Loon reference teaches substituting bytes, not appending data streams. Accordingly, the rejection of claim 1 is improper and should be reversed.

Further, as the Examiner acknowledges, Van Loon reference does not teach delimiting data segments using beginning and end tags. The Examiner argues, however, that these limitations are taught by the Sheedy reference, and that it would be obvious to modify the Van Loom reference to beginning and end tags "in order to utilize the module or object code to record the version of streaming data."

As the Applicants have pointed out above in section VIII.A.2, Van Loon reference is silent as to how the entity receiving the data stream from the data translation devices 7 and 38 handles the incoming data. When analyzing data entering the translators 7 and 38, however, Van Loon teaches two means by which the protocol of the incoming data can be determined. They are (1) by examining the header (bytes A,B), or by looking at the remaining portion of the message (C, D, E, F, G, and H). Van Loon teaches examining a single header *or* the remaining information data, and hence *teaches away* from the Applicants' method of using beginning and end tags in the data stream itself.

Claims 19 and 29 include limitations analogous to those of claim 1, and is patentable for the same reasons.

2. Claims 13, 24, and 30 are Patentable over the Cited References

The Final Office Action also rejected claims 13 and 24 using the Van Loon and Sheedy references. Claim 13 includes the following limitations:

> *receiving a first stream of data according to a first version of the streaming protocol;*
> *if the selected version of the streaming protocol is not the first version of the streaming protocol, sequentially receiving additional streams of data according to each subsequent version of the streaming protocol up to and including the selected version; and*

testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the selected version.

Whatever can be said about the Van Loon reference, one thing is certain. It discloses precious little about how the data stream with data from both protocol versions is handled once received. If such a disclosure is explicit in the Van Loom reference, the Applicants have been unable to find it. Further, the Applicants do not believe that such limitations are inherently disclosed. Inherency "may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient." *Continental Can Co. v. Monsanto Co.*, 948 F.2d 1264, 1269 (Fed. Cir. 1991). Instead, to establish inherency, the extrinsic evidence "must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." *Continental Can Co.*, 948 F.2d at 1268. None of the foregoing steps are necessary to practice the Van Loon reference. In fact, the Van Loon reference teaches away from such steps. Accordingly, the rejection of claim 13 should be reversed.

Claims 24 and 30 include limitations analogous to those of claim 13 and is patentable on this basis.

3. Claim 28 is Patentable over the Cited References

Claim 28 includes limitations analogous to those of both claim 1 and claim 13 and is patentable on this basis.

4. Dependent Claims

Dependent claims 2-8, 10-12, 14-18, 20-21, 23, and 25-27 include the limitations of the claims dependent thereon and are patentable on this basis. The dependent claims also include limitations that render them even more remote from the Van Loon and Sheedy references.

For example, with respect to claim 11, the Examiner suggests that the following limitations

determining whether the data segment is stored in a current context for the data stream;

if so, transmitting an alias tag in lieu of the data segment; and

if not, storing the data segment in the current context.

is disclosed in the following portion of the Sheedy reference:

> Referring now to FIG. 10, the first step 100 is to build a list of the module names to include and exclude from the merge. In the present example, module E is included in the merge. Next, the merge list is tested 102 to determine whether there are more modules to merge. If no, then the program branches to a closing label A 103. If yes, the LN file for module E. and VH files for the FROM and TO variants are opened 104 and a new VH file (RVH) is created 106 for the results of the merge.

The Applicants are at a loss to explain how the steps of claim 11 are disclosed in the foregoing passage.

IX. CONCLUSION

In conclusion, independent claims 1, 13, 19, 24, 29, and 30 of the present application recite novel features that are not found in or suggested by the cited references. In addition, claims 2-8, 10-12, 14-18, 20-21, 23, and 25-27 dependent thereon include additional novel features and are even more remote from the teachings of the cited references. As a result, the Applicants respectfully request the allowance of the present application.

Respectfully submitted,

Mark E. Davis et al.

By their attorneys,

GATES & COOPER

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: May 21, 2001

By: _____

Victor G. Cooper
Reg. No.: 39,641

VGC/io
(30571.77US01)

APPENDIX

THE CLAIMS ON APPEAL (as finally amended)

1. A method of transmitting a data segment in a data stream using a write module which implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:

(a) outputting a first stream of data according to a first version of the streaming protocol;

(b) sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

(c) delimiting the data segment in the data stream using begin and end tags.

2. The method of claim 1, further comprising the step of receiving the data segment from a data stream using a read module of the type which implements a second selected one of the plurality of versions of the streaming protocol, the receiving step including the steps of:

receiving the first stream of data;

if the second selected version is earlier than the first selected version, receiving each additional stream of data according to each subsequent version of the streaming protocol up to and including the second selected version, and disregarding any remaining data in the data segment;

if the second selected version is equal to or later than the first selected version, sequentially receiving the additional streams of data according to each subsequent version of the streaming protocol up to and including the second selected version; and

testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the second selected version.

3. The method of claim 2, wherein the data segment is an object.

4. The method of claim 3, wherein the data segment includes all of the data necessary to reconstruct the object; whereby the data stream is serial.

5. The method of claim 3, wherein the testing step includes the step of initializing object data that is not received from the data stream to a default value.

6. The method of claim 3, further comprising the steps of:

transmitting an object type for the data segment; and

receiving the object type, including allocating and initializing an object when receiving the data segment based upon the object type.

7. The method of claim 2, wherein the read and write modules are resident on the same computer.

8. The method of claim 2, wherein the read and write modules are resident on separate computers.

10. The method of claim 1, wherein no additional tags are embedded in the data segment between the begin and end tags.

11. The method of claim 1, further comprising the steps of:

determining whether the data segment is stored in a current context for the data stream;

if so, transmitting an alias tag in lieu of the data segment; and

if not, storing the data segment in the current context.

12. The method of claim 1, wherein the data stream is a non-random access data stream.

13. A method of receiving a data segment from a data stream using a read module which implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:

(a) receiving a first stream of data according to a first version of the streaming protocol;

(b) if the selected version of the streaming protocol is not the first version of the streaming protocol, sequentially receiving additional streams of data according to each subsequent version of the streaming protocol up to and including the selected version; and

(c) testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the selected version.

14. The method of claim 13, further comprising the step of, if the end of the data segment has not been detected upon receiving the additional stream of data according to the selected version, disregarding any remaining data in the data segment.

15. The method of claim 14, further comprising the step of storing the data segment in a current context, including any disregarded data therefrom.

16. The method of claim 13, wherein the data segment is an object.

17. The method of claim 16, wherein the testing step includes the step of initializing object data that is not received from the data stream to a default value.

18. The method of claim 16, further comprising the steps of:

receiving an object type for the data segment; and

allocating and initializing an object based upon the object type to build the object from the streams of data in the data segment.

19. A computer system that transmits data segment in a data stream, the computer system comprising a write module that implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, and that outputs the data segment in the data stream, wherein:

(a) the write module comprising means for outputting a first stream of data according to a first version of the streaming protocol;

(b) the write module comprising means for sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

(c) the write module comprising means for delimiting the data segment in the data stream using begin and end tags.

20. The computer system of claim 19, wherein the data segment is an object.

21. The computer system of claim 19, wherein the write module further comprises means for transmitting an object type for the data segment.

23. The computer system of claim 19, wherein the write module further comprises means for transmitting an alias tag in lieu of the data segment if the data segment is stored in a current context for the data stream.

24. A computer system that receives a data segment from a data stream, the computer system comprising a read module that implements a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, and that receives the data segment from the data stream, wherein the read module comprises:

(a) means for receiving a first stream of data according to a first version of the streaming protocol;

(b) means for sequentially receiving additional streams of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

(c) means for testing whether an end of the data segment has been detected, and if so, for terminating reception of the data segment prior to receiving the additional stream of data according to the selected version prior to receiving each additional stream of data.

25. The computer system of claim 24, wherein, if the end of the data segment has not been detected upon receiving the additional stream of data according to the selected version, the read module disregards any remaining data in the data segment.

26. The computer system of claim 24, wherein the data segment is an object.

27. The computer system of claim 26, wherein the read module comprises means for receiving an object type for the data segment and for allocating and initializing an object based upon the object type to build the object from the streams of data in the data segment.

28. A computer system comprising first and second computers that transmit a data segment in a data stream from the first computer to the second computer, the first computer comprising means for implementing a first selected one of a plurality of versions of a streaming protocol, and the second computer comprising means for implementing a second selected one of the plurality of versions of the streaming protocol, wherein the second selected one of the plurality of versions of the streaming protocol is additive to the first selected on of the plurality of versions of the streaming protocol, and wherein:

(a) the first computer includes a write module for transmitting the data segment, wherein the write module outputs a first stream of data according to a first version of the streaming protocol, and if the first selected version is not the first version of the streaming protocol, the write module sequentially appends to the first stream of data additional streams of data according to each subsequent version of the streaming protocol up to and including the first selected version; and

(b) the second computer includes a read module for receiving the data segment from the first computer, wherein the read module receives the first stream of data, wherein if the second selected version is earlier than the first selected version, the read module receives each additional stream of data according to each subsequent version of the streaming protocol up to and including the second selected version, and disregards any remaining data in the data segment, wherein if the second selected version is equal to or later than the first selected version, the read module sequentially receives the additional streams of data according to each subsequent version of the streaming protocol up to and including the second selected version, and wherein, prior to receiving each additional stream of data, the read module detects whether an end of the data segment has been detected, and if so, terminates reception of the data segment prior to receiving the additional stream of data according to the second selected version.

29. A program storage device, readable by a computer system and tangibly embodying one or more programs of instructions executable by the computer system to perform method steps of transmitting a data segment in a data stream in a format based upon a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:

(a) outputting a first stream of data according to a first version of the streaming protocol;

(b) sequentially appending additional streams of data to the first stream of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

(c) delimiting the data segment in the data stream using begin and end tags.

30. A program storage device, readable by a computer system and tangibly embodying one or more programs of instructions executable by the computer system to perform method steps of receiving a data segment from a data stream according to a selected one of a plurality of versions of a streaming protocol wherein each subsequent version of the streaming protocol is additive to a previous version, the method comprising the steps of:

(a) receiving a first stream of data according to a first version of the streaming protocol;

(b) sequentially receiving additional streams of data according to each subsequent version of the streaming protocol up to and including the selected version, if the selected version of the streaming protocol is not the first version of the streaming protocol; and

(c) testing, prior to receiving each additional stream of data, whether an end of the data segment has been detected, and if so, terminating reception of the data segment prior to receiving the additional stream of data according to the selected version.

31. The method of claim 13, wherein the step of testing whether and end of the data segment has been detected comprises the step of testing for a premature end tag and terminating the reception of the data segment when a premature end tag is received.

32. The program storage device of claim 30, wherein the step of testing whether and end of the data segment has been detected comprises the step of testing for a premature end tag and terminating the reception of the data segment when a premature end tag is received.